GUÍA TEÓRICO PRÁCTICA 2.

Parte I: Matrices, Listas y Data Frames

Matrices

Una matriz es un arreglo bidimensional. Una forma de crear una matriz es generando un vector y aplicándole la función matrix(). Por defecto S-PLUS llena la matriz con los valores del vector por columna, primero la primera columna, luego la segunda, etc). Para llenar una matriz por fila hay que utilizar el argumento "byrow=T".

1. Generación de matrices

1.a Analice los siguientes ejemplos

```
> x<-1:10
> matrix(x,nrow=2,ncol=5)
> matrix(x,2,5,byrow=T)

> x<-1:4
> matrix(x,3,8)
```

1.b La función **scan** permite entrar datos en forma interactiva en la ventana de comandos. Ingrese 15 datos en forma interactiva, ingrese una cantidad variable de datos por fila

```
> y <- scan()
```

¿Que indica el número que aparece al comienzo de cada fila?
Interrumpa el ingreso de datos con “dos veces enter”.

1.c Ingrese una matriz por columnas, por ejemplo:

```
> y2 <- matrix(scan(),ncol=3)
1: 1
2: 1 2 3 4 5
7: 1 2 3
10:

> y2
     [,1] [,2] [,3]
[1,]    1    3    1
[2,]    1    4    2
[3,]    2    5    3
```

Vea que tipo de objeto ha creado:

```
> data.class(y2)
[1] "matrix"
```

Construya la misma matriz utilizando la función **c**

```
> y2<- matrix(c(1,1,2,3,4,5,1,2,3),ncol=3)
> y2
     [,1] [,2] [,3]
[1,]    1    3    1
[2,]    1    4    2
```

```
           [3,]    2    5    3
```

b2) Pruebe con

```
matrix(c(1,1,2,3,4,5,1,2,3),4,byrow=T)
matrix(c(1,1,2,3,4,5,1,2,3),ncol=3,byrow=T)
```

2. Atributos de una matriz

Las matrices tienen los siguientes atributos: cantidad de filas, cantidad de columnas, dimensión, longitud y modo. Se accede a los valores de los atributos mediante las funciones: **nrow(), ncol(), dim(), length()** y **mode()** respectivamente**.** El valor de **dim(x)** es **c(nrow(x),ncol(x))** y el valor de **length(x)** es **nrow(x)*ncol(x).**

```
> Matriz.deDatos <- matrix(1:12, ncol = 4)
> dim(Matriz.deDatos)
[1] 3 4
 > length(Matriz.deDatos)
 [1] 12
 > mode(Matriz.deDatos)
 [1] "numeric"
```

Observación: las matrices, igual que los vectores, deben contener elementos de un mismo modo.

Veremos hacia el final de la guía que **dimnames(),** que es un conjunto de nombres para las filas y las columnas.

3. Funciones cbind rbind y t. Otra forma de obtener matrices.

```
> col1<- c(1,2,3)
> col2<- c(4,5,6)
> col3<- c(0,0,0)
> A<- cbind(col1,col2,col3) #pegamos los vectores en columnas
> B<- rbind(col1,col2,col3) #pegamos los vectores en filas
> t(col1)
     [,1] [,2] [,3]
[1,]    1    2    3
> t(t(col1))
     [,1]
[1,]    1
[2,]    2
[3,]    3
```

4. Operaciones algebraicas.

Las operaciones algebraicas (+, -, *, /, ^, log(), sqrt(), ..) aplicadas a matrices se realizan **componente a componente**, igual que para los vectores.

```
> x<-matrix(1:10,2,5,byrow=T)
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
> x*2
     [,1] [,2] [,3] [,4] [,5]
[1,]    2    4    6    8   10
```

```
[2,]   12   14   16   18   20
> exp(x)
           [,1]        [,2]       [,3]       [,4]       [,5]
[1,]   2.718282    7.389056   20.08554   54.59815   148.4132
[2,] 403.428793 1096.633158 2980.95799 8103.08393 22026.4658
```

Si las matrices no tienen igual dimensión, la de menor dimensión es expandida.

```
  > x
       [,1] [,2] [,3] [,4] [,5]
  [1,]    1    2    3    4    5
  [2,]    6    7    8    9   10
> x+c(0,1)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    7    8    9   10   11
```

5.1 Operaciones matriciales

Multiplicación matricial: "%*%". Transpuesta: t(x) Inversa: solve(x)

```
  > x
       [,1] [,2] [,3] [,4] [,5]
  [1,]    1    2    3    4    5
  [2,]    6    7    8    9   10
> y
     [,1] [,2]
[1,]    1    0
[2,]    0    2
> t(x)%*%y
     [,1] [,2]
[1,]    1   12
[2,]    2   14
[3,]    3   16
[4,]    4   18
[5,]    5   20
> solve(y)
     [,1] [,2]
[1,]    1  0.0
[2,]    0  0.5
```

5.2 Mire el help para más detalles sobre ("%*%").

6. Acceso a los elementos de las matrices

Los corchetes también permiten el acceso a los elementos de una matriz.
Una coma dentro de los corchetes separa los índices de filas y columnas. Analice los siguientes ejemplos.

```
> x <- matrix(1:15, 3, byrow = T)
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
```

```
> x[2, 3]
[1] 8
> x[1:2, 1:3]
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    6    7    8
> x[, 1]
[1]  1  6 11
> x[, 2:3]
     [,1] [,2]
[1,]    2    3
[2,]    7    8
[3,]   12   13
> x[, -4]
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    5
[2,]    6    7    8   10
[3,]   11   12   13   15
```

Si no se pone la coma dentro de los corchetes la matriz es tratada como un vector

```
> x[3]
[1] 11
> x[2]
[1] 6
> x[3:7]
[1] 11  2  7 12  3
> x[x == 4] <- NA
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3   NA    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
```

Las matrices en S-plus **son** vectores con atributos adicionales.

7. Operaciones por filas o columnas.

La función apply permite realizar un cálculo por fila ó por columna
Si x es una matriz de n x p entonces

```
> apply(x,2,mean)
```

dará un vector de longitud p (cantidad de columnas) con las medias de las columnas. El argumento 2 en **"apply(x,2,mean)"** indica que el cálculo debe realizarse en la segunda dimensión.

```
> apply(x,1,mean)
```

dará un vector de longitud n (cantidad de filas) con las medias de las filas.

Con la función **apply()** se pueden calcular varianzas, medianas, sumas, productos, ..etc. Casi cualquier función que calcule un valor resumen de un vector**:**

```
> apply(x,2,sum)  # sumas por columnas
> apply(x,2,var)  # varianzas por columnas
```

```
> sqrt(apply(x,2,var))   # desvíos estándar por columna
```

Verifique los resultados del siguiente ejemplo

```
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3   NA    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15

> apply(x, 2, mean)
[1]  6  7  8 NA 10

> apply(x, 1, mean)
[1] NA  8 13
```

8. Listas

Una lista es una colección de objetos pegados de manera que pueden ser indicados por un único nombre. Puede utilizarse la función **list()** como en este ejemplo:

```
> x <- matrix(1:4, 4, 7)
> y <- "Esta es la segunda practica de Analisis de Datos"
> z <- c(T, F, T, NA)
> primeralista <- list(x, y, z)

> primeralista
[[1]]:
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    1    1    1    1    1    1    1
[2,]    2    2    2    2    2    2    2
[3,]    3    3    3    3    3    3    3
[4,]    4    4    4    4    4    4    4

[[2]]:
[1] "Esta es la segunda practica de Analisis de Datos"

[[3]]:
[1]  T  F  T NA
```

“primeralista” tiene tres elementos. El primero es una matriz, el segundo es una cadena de caracteres, y el tercero es un vector lógico. Se puede acceder a los elementos individuales de la lista utilizando doble corchetes:

```
> primeralista[[1]]
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    1    1    1    1    1    1    1
[2,]    2    2    2    2    2    2    2
[3,]    3    3    3    3    3    3    3
[4,]    4    4    4    4    4    4    4
> primeralista[[2]]
[1] "Esta es la segunda practica de Analisis de Datos"
> primeralista[[3]]
[1]  T  F  T NA
```

Pueden asignarse nombres a cada elemento de la lista

```
> names(primeralista) <- c(“Matriz”, “Leyenda”,”Vector Logico”)
```

o declarar los nombres cuando la lista es creada

```
> primeralista <- list(Matriz=x, Leyenda=y,Vector Logico=z)

> primeralista
$Matriz:
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    1    1    1    1    1    1    1
[2,]    2    2    2    2    2    2    2
[3,]    3    3    3    3    3    3    3
[4,]    4    4    4    4    4    4    4

$Leyenda:
[1] "Esta es la segunda practica de Analisis de Datos"

$"Vector Logico":
[1]  T  F  T NA
```

Para referirse a un elemento de la lista tenemos tres formas, dos por su nombre y la inicial por el número de la componente:

```
> primeralista$Leyenda
[1] "Esta es la segunda practica de Analisis de Datos"

> primeralista[["Leyenda"]]
[1] "Esta es la segunda practica de Analisis de Datos"

> primeralista[[2]]
[1] "Esta es la segunda practica de Analisis de Datos"
```

No es necesario usar el nombre completo

```
> primeralista[["Vector"]]
[1]  T  F  T NA
> primeralista$Vector
[1]  T  F  T NA
```

La longitud de la lista es la cantidad de elementos de la lista. Los nombres de la lista constituyen un vector de caracteres de igual longitud que la lista

```
> length(primeralista)
[1] 3
> names(primeralista)
[1] "Matriz"        "Leyenda"       "Vector Logico"
```

9. Agregar o eliminar elementos de una lista.

9.a Cree la lista `Provincias` **con cuatro componentes cuyos nombres son “Buenos Aires” “Cordoba” “Santa Fe” “Capital Federal”**

9.b Elimine la componentes "Capital Federal" y "Buenos Aires"

```
> Provincias[-3] # elimina transitoriamente Capital Federal
> Provincias2 <- Provincias[-3] #nueva lista sin la componente Capital
Federal

> Provincias[-1] # elimina transitoriamente Buenos Aires
```

Verifique lo comentado

```
> Provincias$Cordoba <- NULL  # elimina en forma permanente Cordoba de
                                  # la lista Provicnias
                                   # funciona si el nombre tiene una
                                    # sola componente


> Provincias$Buenos <- NULL #  vea qué pasa
```

Verifique lo comentado

La lista quedó con 2 componentes. Agregue las componentes con nombres "Santiago del Estero" "Corrientes" y "Misiones" y valores:

```
> Provincias$"Santiago del Estero" <- "menor que 185"
> Provincias$Corrientes <- "185-200"
> Provincias$Misiones <- "mayor que 200"
```

Verifique que la lista tiene ahora 5 componentes.

10. Nombres de las filas y columnas de una matriz.

Una matriz tiene un atributo llamado **dimnames()**.

Si x es una matriz entonces dimnames(x) es una lista con dos elementos. El primer elemento es un vector de cadenas de caracteres de longitud nrow(x) que contiene los nombres de las filas. El segundo elemento de dimnames(x) es un vector conteniendo los nombres de las columnas, es un vector de cadenas de caracteres de longitud ncol(x)

Podemos agregar nombres a las filas y columnas de una matriz x creando esa lista y asignandosela a dimnames(x).

```
> notas <- c(100, 99, 100, 97, 54, 26)
> notas <- matrix(notas, 3, 2, byrow = T)
> dimnames(notas) <- list(c("Juan", "Susana",   "Sebastian"),
   c("Parcial", "Final"))
> notas
          Parcial Final
     Juan     100    99
   Susana     100    97
Sebastian      54    26
```

Si se quiere dar nombres unicamente a las columnas ó a las filas utilice NULL:

```
> dimnames(notas) <- list(NULL, c("Parcial", "Final"))
> notas
     Parcial Final
[1,]     100    99
[2,]     100    97
[3,]      54    26

> dimnames(notas) <- list(c("Juan", "Susana", "Sebastian"), NULL)
> notas
          [,1] [,2]
     Juan  100   99
   Susana  100   97
Sebastian   54   26
```

Cuando las filas o las columnas tienen nombres, es posible referirse a filas o columnas específicas por su nombre:

```
> dimnames(notas) <- list(c("Juan", "Susana", "Sebastian"),
   c("Parcial",
   "Final"))
> notas
          Parcial Final
     Juan     100    99
   Susana     100    97
Sebastian      54    26
> notas["Susana",  ]
 Parcial Final
     100    97
> notas[, "Final"]
 Juan Susana Sebastian
   99     97        26
```

11. Data Frames

Un data frame (marco de datos) está basado en la idea de que muchos conjuntos de datos en estadística pueden organizarse en un arreglo rectangular con las filas representando unidades muestrales y las columnas representando variables. Un data frame tiene un aspecto semejante a una matriz pero es más general ya que las columnas pueden tener diferentes modos de almacenamiento. En un data frame una columna puede contener números, otra valores lógicos (T o F) y otra columna cadena de caracteres. Todas las columnas deben tener la misma longitud.

```
> x1 <- c(100, 99, 100, 20)
> x2 <- c(20, 19, 19, 10)
> x3 <- c("A", "A", "A", "C")
> notas <- data.frame(x1, x2, x3)
> notas
   x1 x2 x3
1 100 20  A
2  99 19  A
3 100 19  A
4  20 10  C
```

Si en alguna variable falta algún valor, debe completarse con NA, para que todos los vectores que forman las columnas tengan igual longitud.

```
> x1 <- c(100, 99, 100, 20)
> x2 <- c(20, 19, 19, NA)
> x3 <- c("A", "A", "A", "C")
> notas <- data.frame(x1, x2, x3)
> notas
   x1 x2 x3
1 100 20  A
2  99 19  A
3 100 19  A
4  20 NA  C
```

12. Data frames y matrices

Las data frames, como las matrices, tienen los atributos "dim", "nrow", "ncol" y "dimnames" attributes:

```
> dim(notas)
[1] 4 3
> nrow(notas)
[1] 4
> ncol(notas)
[1] 3
> dimnames(notas)
[[1]]:
[1] "1" "2" "3" "4"

[[2]]:
[1] "x1" "x2" "x3"
```

Se pueden cambiar las leyendas de las filas y columnas y igual que con las matrices

```
> dimnames(notas) <- list(c("Susana", "Juan", "Diago", "Micaela"), c(
   "trabajo", "examen", "concepto"))
> notas
        trabajo examen concepto
 Susana     100     20        A
   Juan      99     19        A
  Diago     100     19        A
Micaela      20     NA        C
```

Como con las matrices, se puede seleccionar subconjuntos con los corchetes:

```
> notas[1:2,]
      trabajo examen final
Susana     100   20     A
 Juan       99   19     A

> notas[,1]
[1] 100  99 100  20

> notas[,"trabajo"]
[1] 100  99 100  20
```

```
> notas[,-1]
        examen final
  Susana  20     A
   Juan   19     A
  Diago   19     A
Micaela   10     C
```

"notas[,1]" resulta en un vector

"notas[1,]", resulta en un data frame con una fila, pues puede tener varios modos de almacenamiento.

La función data.frame() puede utilizarse para **transformar una matriz en un data frame**. Si la matriz no tiene leyendas de filas y columnas entonces data.frame() le dará "1", "2", ... a las filas y "X1", "X2", etc. a las columnas. Si tienen nombres, se mantienen.

```
> v<-matrix(rnorm(12),4,3)

> data.frame(v)
          X1         X2         X3
1  1.3289191  1.5826525  0.5284931
2 -0.5834504  0.7756624 -0.5783929
3  0.1294323 -0.1010787 -0.7104860
4  0.9805573 -0.8096563  0.1749112
```

13. Data frames como listas

Un data frame es una lista en la que cada columna es un elemento de la lista. Como las listas los elementos pueden ser referidos por sus posiciones utilizando dobles corchetes ([[]]) o utilizando sus nombres

```
> notas[[1]]
[1] 100  99 100  20
> notas[[3]]
[1] A A A C
> notas$final
[1] A A A C
```

Los atributos longitud y nombres ("length", "names") son los mismos que en una lista

```
> length(notas)
[1] 3
> names(notas)
[1] "trabajo" "examen"     "final"
```

Un data frame es un tipo especial de lista en la cual todos sus elementos son vectores de la misma longitud.

Parte II. Histogramas, Diagramas de tallo hoja y Estimación de densidades

1-1Histogramas

Trabajaremos con un conjunto de datos existentes en “la librería” del S-plus
Para obtener información sobre el archivo de datos hacemos

```
help(geyser)
```

```
(en R help(faithful), si trabaja en R reemplace geyser por faithful)
```

a) Construya un histograma para las distintas variables (verifique el nombre de las variables en el R)

```
hist(geyser$waiting)
hist(geyser$duration)
hist(geyser$duration, density =-1)
```
#obtenemos barras sin relleno

b) Mire el help para obtener el histograma de acuerdo a los criterios de Scott y Freedman & Diaconis

c) Cambie el valor inicial del histograma y la longitud del intervalo de clase.(mire el help)

1-2 Un **polígono de frecuencias** se obtiene uniendo los puntos medios de las barras de un histograma.

```
a) hist(geyser$waiting,plot=F)
$breaks:
 [1]  40  45  50  55  60  65  70  75  80  85  90
[12]  95 100 105 110

$counts:
 [1]  2 26 29 25 17 10 34 59 44 35 14  3  0  1
```

Para hallar los puntos medios podemos hacer
X1= `hist(geyser$waiting,plot=F)$breaks[-1]`
X2= `hist(geyser$waiting,plot=F)$breaks[-15]`
Pm= (X1+X2)/2
lines (pm, `hist(geyser$waiting,plot=F)$counts)****` superpone el poligono sobre el histograma ya graficado

b) Repita para **`geyser$duration`**

1-3. Otras distribuciones.

a) Grafique, en la misma pantalla, histogramas con muestras de la distribución t con 1, 2, 3, 10, 30 y 50 grados de libertad. Idem para la distribución Chi-cuadrado.
b) Construya un histograma con la mezcla de una muestra de tamaño 100 de una N(0,1) y otra muestra de tamaño 100 de una N(3,1)

2- Diagramas tallo hoja.

Un diagrama tallo-hoja se obtiene mediante la instrucción **`stem`**.

a1) Pruebe con los datos de **`swiss.fertility`** las diferentes opciones de **`stem`**. Utilice el **`help`** para hallar las diferentes opciones de **`stem`**.
a2) Construya diagramas tallo-hoja de todas las variables que aparecen en **`swiss.x.`** Utilice el **`help`** para entender de que se tratan las variables. El diagrama de la quinta columna se obtiene:

```
> stem(swiss.x[,5])
```

b) Estudie mediante un diagrama tallo-hoja las mediciones del cobre en harina integral.
c) Pruebe las dos opciones siguientes y compare.

```
> stem(abbey)
> stem(abbey,scale=-1)
```

Observe que cambia la posición del punto decimal (`colon`).

d) Estudie mediante diagramas de tallo-hoja y diferentes histogramas los datos de muertes mensuales por enfermedades de pulmón dados en **`mdeaths`** (hombres) **`fdeaths`** (mujeres). Estudie también los dos conjuntos de datos juntos, utilice histogramas con la misma escala para poder compararlos.
El `help` nos dice:

> Monthly Deaths from Lung Diseases in the UK
>
> Monthly deaths from bronchitis, emphysema and asthma in the UK, 1974-1979, both sexes (deaths), males (mdeaths) and females (fdeaths)
>
> SOURCE:
> P.J. Diggle (1990) Time Series: A Biostatistical Introduction Oxford, table A.3

e) Explique qué significa la línea de profundidad de un diagrama tallo-hoja. Utilice la profundidad dada en los diagramas para hallar la mediana, los cuartos y los cuartiles de los datos considerados.

3. 1Gráficos de funciones de densidad.

a) Pruebe el siguiente comando para obtener el gráfico de la densidad N(0,1).

```
> plot(u<-seq(-3,3,length=50),dnorm(u),type="l",xlab="",
  ylab="", sub="Funcion de densidad Normal")
```

b1) Podemos obtener la función de ("densidad") probabilidad puntual de una **`Bi(10,0.1)`** con

```
> barplot(dbinom(c(0:10),10,0.1),sub= "Bi(10,0.1)")
```

b2) Utilice el `help` para modificar los parámetros gráficos de `barplot`, también se pueden utilizar con la función `hist`.

c) Repita a) y b1) para diferentes distribuciones y con diferentes parámetros..

3.2 Estimación de densidades.

a) La función **`density`** le permite obtener un estimador de densidad de un conjunto de datos. Utilice el **`help`**.

b) Para los datos de los ejercicios 4, 5 y 6 obtenga estimadores de densidad, con distintos tamaños de ventanas y grafíquelos. Pruebe modificar las distintas opciones del comando `density`.

c) Superponga la curva de densidad a los histogramas.